Pemodelan Algoritme Backtracking dalam Menyelesaikan Permainan 2048 Deterministik

Mohammad Sheva Almeyda Sofjan 13519018
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519018@std.stei.itb.ac.id

***Abstrak*— Permainan 2048 merupakan puzzle atau permainan teka-teki berbasis kotak geser (*sliding tile puzzle*) dimainkan dalam mode *single-player* (memiliki pemain tunggal), di mana pemain bermain pada grid berukuran N×N, dengan kotak geser yang ditandai dengan label berupa bilangan bulat perpangkatan 2 mulai dari 2 hingga bilangan perpangkatan 2 terbesar yang dapat diakomodasi oleh permainan, yang bertujuan untuk mencapai kotak dengan bilangan berlabel 2048. Pada makalah ini akan dibahas pemodelan algoritme backtracking dalam menyelesaikan permainan 2048 yang deterministik, dengan memanfaatkan pohon ruang status yang dibentuk seiring permainan berlangsung.**

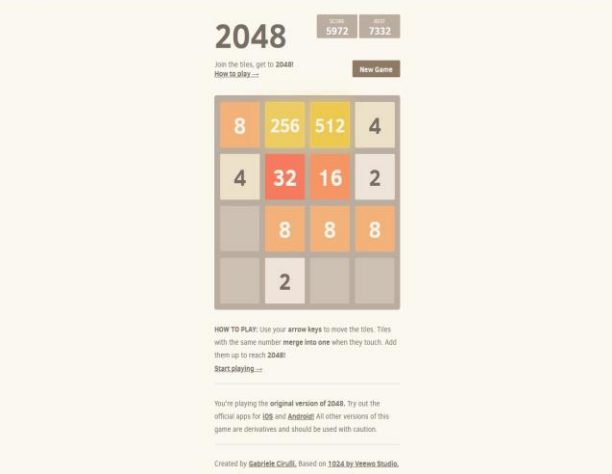
Kata Kunci—2048, Algoritme, Backtracking, Pohon

I. PENDAHULUAN

Permainan 2048 merupakan puzzle atau permainan teka-teki berbasis kotak geser (*sliding tile puzzle*) yang diciptakan oleh pengembang web asal Italia bernama Gabriele Cirulli yang dipublikasikan pada bulan Maret 2014.

Permainan ini dibuat berdasarkan permainan serupa, yaitu 1024 dan *Threes* yang dirilis satu bulan lebih awal dari permainan 2048 ini. Permainan ini awalnya dipublikasikan pada GitHub pengembang kemudian dirilis pada platform Web, Android, iOS, dan beberapa platform lainnya. Pada awal rilisnya, permainan ini mendapatkan tanggapan positif dari berbagai kalangan, sebagai permainan yang adiktif sehingga menjadi cukup populer pada masanya. Sebagai referensi, tautan resmi permainan ini adalah https://play2048.co/ . [4]

Permainan 2048 ini merupakan permainan dengan mode *single-player* (memiliki pemain tunggal), di mana pemain bermain pada grid berukuran N×N, biasanya berukuran 4×4, dengan kotak geser yang ditandai dengan label berupa bilangan bulat perpangkatan 2 mulai dari 2 hingga bilangan perpangkatan 2 terbesar yang dapat diakomodasi oleh permainan (untuk kasus grid berukuran 4×4, bilangan terbesar yang mungkin adalah 131.072, atau ekuivalen dengan 2^{17}) .



Gambar 1. Permainan 2048

Tangkapan layar diambil dari https://play2048.co/

Tujuan utama dari permainan ini adalah untuk mencapai bilangan 2048 yang dapat dibentuk dari hasil pergeseran kotak yang mampu menyatu dengan kotak lainnya bila merupakan kotak dengan label yang memiliki nilai bilangan yang sama, sehingga pemain dinyatakan memenangkan permainan. Bila hal tersebut belum cukup, maka pada beberapa skenario permainan pemain dapat melanjutkan permainan untuk mencapai skor setinggi-tingginya hingga tidak terdapat kotak yang mampu digabungkan lagi dengan kotak lainnya, atau pemain dinyatakan kalah ketika itu terjadi.

Penggunan algoritme *backtracking* dalam penyelesaian persoalan terkait permainan 2048 ini adalah untuk menggambarkan *state-state* atau status dari permainan yang mungkin terjadi ketika dilakukan suatu gerakan terhadap permainan (menggeser dan menggabung-gabungkan kotak), yang dapat memberitahu langkah yang diperlukan, banyak langkah minimum dan maksimum yang dilakukan untuk mendapatkan kotak dengan label tertentu dengan ukuran grid permainan tertentu.

II. LANDASAN TEORI

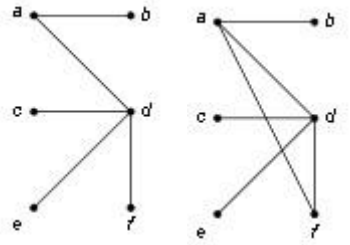
A. *Definisi Algoritme*

Algoritme, menurut Kamus Besar Bahasa Indonesia[1] merupakan prosedur sistematis untuk memecahkan masalah

matematis dalam langkah-langkah terbatas, atau urutran logis pengambilan keputusan untuk pemecahan masalah.

B. Definisi Pohon Ruang Status

Pohon merupakan graf tak berarah terhubung yang tak mengandung sirkuit. Graf G = *(V,E)* dikatakan sebagai pohon apabila merupakan graf sederhana dan tidak mengandung sirkuit.



Gambar 2. Sebuah pohon (kiri) dan bukan pohon (kanan)
Diambil dari
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf pada 10 Mei 2021

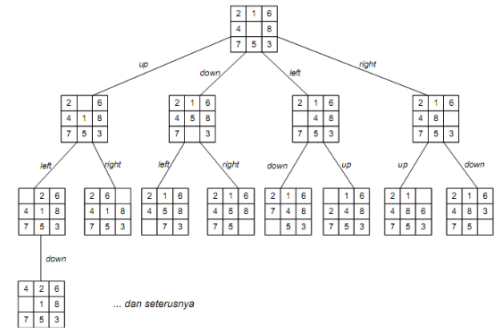
Pohon ruang status (*state space tree)* sendiri adalah pohon yang menggambarkan status-status (skenario) yang dibangkitkan pada suatu persoalan kedalam simpul-simpul pada pohon tersebut.

Akar dari pohon merupakan status awal (*initial state)* dari suatu persoalan, bisa berupa status yang sudah didefinisikan persoalan atau bisa juga berupa status kosong.

Simpul dari pohon merupakan status dari persoalan (*problem state)*, yang menggambarkan keadaan pada persoalan setelah suatu algoritme atau perintah dieksekusi pada simpul orangtua-nya.

Sisi dari pohon, penghubung antara suatu simpul dengan simpul anak-nya memiliki bobot, berupa operator atau perintah yang dipanggil pada suatu simpul agar simpul tersebut "berubah" statusnya menjadi simpul anak-nya, dengan kata lain berguna untuk mengubah status dari suatu persoalan.

Sedangkan daun dari pohon, merupakan status solusi (*solution state)* atau simpul tujuan dari persoalan, yang menggambarkan status akhir (dapat juga berupa status yang "dipangkas") dari suatu persoalan, yang tidak perlu diaplikasikan operator apapun untuk mengubah status nya. Ruang solusi (*solution space)* terdiri atas himpunan semua status solusi dari pohon ruang status.[2]



Gambar 3. Contoh Pohon Ruang Status
Diambil dari
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf pada 10 Mei 2021

C. Algoritme Depth-First-Search

Depth-First-Search (DFS) atau pencarian mendalam merupakan algoritme penelusuran graf yang teknisnya adalah melakukan penelusuran ke semua simpul anak terlebih dahulu hingga semua simpul tetangga pada simpul anak telah dikunjungi, lalu kembali ke simpul yang belum dikunjungi lalu ulangi penelusuran pada anak hingga semua simpul telah dikunjungi. Misal terdapat simpul awal *v* pada suatu graf *G*, G akan ditelusuri secara DFS dengan algoritme sebagai berikut.[2]

Starting point : simpul v

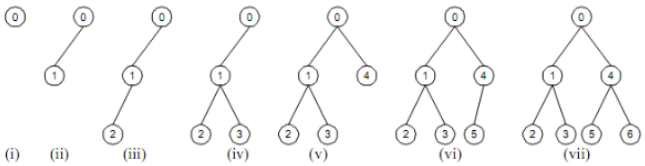
Kunjungi simpul w yang merupakan tetangga dari simpul v

Ulangi DFS mulai dari simpul w.

Saat mencapai suatu simpul u dan semua simpul tetangganya sudah dikunjungi, maka kembali ke simpul terakhir yang sudah dikunjungi sebelumnya dan mempunyai simpul tetangga yang belum dikunjungi

Pencarian diakhiri ketika tidak ada lagi simpul yang bisa dikunjungi dari simpul-simpul yang telah dikunjungi

Penggunaan DFS dalam pembangkitan status pada pohon ruang status digambarkan lewat ilustrasi berikut.



Gambar 4. Pembangkitan Ruang Status - DFS
Diambil dari
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf pada 10 Mei 2021

D. Algoritme Backtracking

Algoritme *backtracking* merupakan algoritme yang merupakan perbaikan dari algoritme *exhaustive search*, karena pada algoritme *backtracking*, hanya status yang mengarah ke solusi yang diekspansi, dan status yang tidak mengarah ke solusi dipangkas atau tidak diekspansi lebih lanjut. Algoritme ini mampu digunakan sebagai metode pemecahan masalah yang cukup efektif, terstruktur, dan sistematis untuk persoalan optimasi maupun non-optimasi.

Algoritme *backtracking* memiliki properti umum yang di antaranya sebagai berikut. [3]

1. Solusi Persoalan

   Umumnya solusi persoalan dalam algoritme ini dinyatakan dalam bentuk vektor dengan *n-tuple*, seperti $X = (x_1,x_2,...,x_n)$, dengan $x_i \in S_i$. Umumnya $S_1=S_2=...=S_n$.

2. Fungsi Pembangkit nilai x_k

   Fungsi pembangkit pada algoritme ini dinyatakan sebagai predikat *T()*, dengan $T(x_1,x_2,...,x_{k-1})$ membangkitkan nilai untuk x_k , yang merupakan komponen dari vektor solusi persoalan

3. Fungsi Pembatas (*bounding function)*

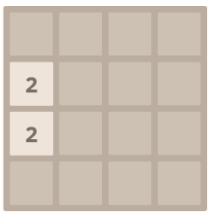
   Fungsi pembatas pada algoritme ini dinyatakan sebagai predikat *B()*, dengan $B(x_1,x_2,...,x_k)$ akan bernilai *true* bila $(x_1,x_2,...,x_k)$ mengarah ke solusi, dengan kata lain tidak melanggar *constraint* persoalan. Jika nilai B bernilai *true*, maka pembangkitan nilai untuk x_{k+1} akan dilanjutkan, sedangkan jika bernilai *false*, maka pembangkitan nilai untuk x_{k+1} tidak dilanjutkan dan $(x_1,x_2,...,x_k)$ dibuang.

Solusi pada algoritme ini dimodelkan ke dalam struktur pohon berakar, tepatnya pohon ruang status, yang mengandung semua kemungkinan solusi dari persoalan sebagai ruang solusi dari pohon ruang status. Rincinya, tiap simpul pohon menyatakan status dari persoalan, sedangkan sisi-sisi dilabeli nilai-nilai pada vektor solusi yaitu x_i. Lintasan dari akar ke daun menyatakan solusi yang mungkin bagi persoalan.

Untuk mencari solusi, perlu dilakukan pembangkitan simpul-simpul status pada pohon sehingga menghasilkan lintasan dari akar ke daun. Dalam algoritme ini, digunakan *depth-first-search* (DFS) sebagai algoritme untuk pembangkitan simpul. Simpul-simpul yang sudah dibangkitkan disebut simpul hidup, dan simpul hidup yang sedang diperluas (diekspansi) dinamakan simpul ekspan. Setiap kali simpul ekspan diperluas, maka lintasan yang dibangun oleh simpul tersebut bertambah panjang, dan jika lintasan yang sedang dibangun tidak mengarah solusi (melanggar fungsi pembatas), maka simpul ekspan tersebut dimatikan sehingga menjadi simpul mati. Jika pembentukan lintasan berhenti dengan adanya simpul mati, maka dilakukan *backtrack* untuk kembali ke simpul orangtua-nya, lalu teruskan dengan membangkitkan simpul anak lainnya. Pencarian dihentikan ketika sudah mencapai simpul tujuan.

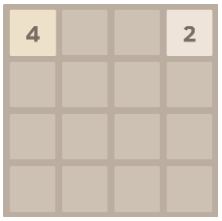
E. Aturan Permainan 2048

Pada awal permainan, pemain disediakan grid berukuran N×N, dengan terdapat satu atau dua grid yang sudah terisi dengan kotak yang memiliki label bilangan (biasanya angka 2). Normalnya, cara permainan menentukan lokasi kotak *starter* tersebut adalah dengan cara dirandomisasi lokasinya pada grid.



Gambar 5. Status awal permainan ukuran 4x4
Tangkapan layar diambil dari https://play2048.co/

Yang dapat pemain lakukan adalah, menggerakkan kotak-kotak tersebut ke arah atas, bawah, kiri, dan kanan sehingga mencapai objektif yang ingin dicapai (yaitu mendapatkan kotak dengan label 2048). Ketika kotak digerakkan, jika terdapat dua kotak bertetanggaan yang memiliki label dengan nilai yang sama, maka dua kotak tersebut akan bergabung menjadi hasil jumlah dua kotak tersebut. Dalam kasus menggunakan status awal permainan seperti pada gambar (4), ketika kotak digerakkan kearah atas, maka akan menghasilkan kotak berlabel 2 + 2 = 4 pada grid kosong teratas, dan menghasilkan status permainan seperti berikut.



Gambar 6. Permainan digerakkan ke atas
Tangkapan layar diambil dari https://play2048.co/

Perlu diperhatikan bahwa, kotak berlabel 2 pada kanan atas grid dimunculkan ketika kotak digerakkan dan masih terdapat grid yang kosong, dan lokasi kemunculan (juga nilai label) dari kotak tersebut juga dirandomisasi, pada kasus ini kebetulan berada pada pojok kanan atas grid. Selain kotak berlabel 2, permainan juga dapat memunculkan kotak berlabel 4, dengan probabilitas kemunculan yang biasanya lebih rendah dari kotak dengan label 2.

Penggabungan 2 kotak pada permainan ini juga memiliki prioritas, yaitu diurutkan sesuai dengan arah pergerakan kotak. Pada kasus dibawah ini, ketika permainan digerakkan ke arah kiri, maka prioritas penggabungan akan condong ke arah kiri, yaitu mengutamakan penggabungan dua kotak angka “4” yang paling kiri, dan angka “4” setelahnya bergeser ke kiri, sehingga [16 4 4 4] menjadi [16 8 4 null]. Sebagai perbandingan, jika terdapat 4 kotak berlabel sama pada satu baris, semisal [4 4 4 4], jika digeser ke arah kanan, maka akan menjadi [null null 8 8].

Gambar 7. Prioritas penggabungan (kiri – sebelum, kanan – sesudah)
Tangkapan layar diambil dari https://play2048.co/

Permainan berakhir ketika semua grid sudah penuh dengan kotak berlabel dan tidak ada kotak bertetanggaan yang mampu digabung satu sama lain. Pemain dinyatakan menang ketika berhasil mendapatkan kotak tujuan pada grid, semisal 2048. Walaupun begitu, normalnya pemain masih dapat melanjutkan permainan hingga semua grid penuh dan tidak ada kotak bertetanggaan yang mampu digabung satu sama lain . Pemain dinyatakan kalah ketika grid penuh dan tidak ada kotak bertetanggaan yang mampu digabung satu sama lain tanpa mendapatkan kotak tujuan.

Gambar 8. Status akhir permainan(kiri – menang, kanan – permainan berakhir).
Tangkapan layar diambil dari https://play2048.co/

III. PEMBAHASAN

A. *Batasan Persoalan*

Permainan 2048 yang digunakan sebagai lingkup persoalan pada makalah ini bukan permainan 2048 yang standar, yang biasa dimainkan. Permainan 2048 yang biasa dimainkan memiliki sifat non-deterministik pada pemunculan kotak baru ketika awal permainan dan setelah dilakukan langkah yang valid. Pada permainan 2048 non-deterministik tersebut pemunculan kotak baru memiliki nilai dan lokasi pada grid kosong yang dirandomisasi, dengan probabilitas munculnya kotak dengan angka 2 lebih tinggi dibanding kotak dengan angka 4.

Sedangkan pada permainan 2048 yang digunakan pada lingkup makalah ini, adalah permainan 2048 yang deterministik, yaitu permainan 2048 standar namun elemen-elemen non-deterministiknya dihilangkan. Alhasil, pemunculan kotak baru memiliki nilai yang tetap, yaitu kotak dengan angka 2, juga lokasi yang mengikuti urutan prioritas kotak kosong pada grid, sesuai dengan indeks pemodelan matriks(i,j) terhadap grid, semakin kecil nilai indeks i,j semakin besar prioritas pemunculan kotak pada sel tersebut.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Gambar 9. Prioritas pemunculan kotak baru dengan angka 2, semakin kecil nilai pada kotak semakin besar prioritas.
Sumber : dokumen penulis

B. *Aplikasi Algoritme*

Pada sub-bab ini, algoritme *backtracking* akan digunakan untuk membangkitkan pohon ruang status dari permainan, di mana pohon ruang status akan tumbuh ketika dilakukan gerakan terhadap permainan. Pemodelan algoritme *backtracking* dalam menyelesaikan permainan 2048 deterministik adalah sebagai berikut.

1. Solusi Persoalan

   Solusi persoalan pada persoalan ini adalah *sequence* atau larik yang berisi daftar gerakan yang dilakukan terhadap papan permainan hingga dicapai status akhirnya, yaitu mendapatkan kotak dengan label tujuan (dapat bervariasi tergantung *test* yang akan dilakukan nantinya).

2. Fungsi Pembangkit

   Fungsi pembangkit pada persoalan ini adalah algoritme yang mensimulasikan pergerakan kotak-kotak pada permainan hingga didapat solusi yang diinginkan, lalu memasukkannya ke dalam larik solusi.

3. Fungsi Pembatas

   Fungsi pembatas pada persoalan ini adalah fungsi yang membatasi ekspansi pohon ruang status, dengan melakukan *pruning* atau pemangkasan apabila gerakan yang dilakukan tidak menghasilkan perubahan status pada permainan (terdapat simpul anak = simpul orangtua) atau menghasilkan status permainan selesai (dapat berupa menang maupun kalah)

Sebagai gambaran, berikut adalah sistematika aplikasi algoritme *backtracking* dalam menyelesaikan permainan 2048 deterministik.

1. Inisialisasi permainan, tempatkan dua kotak pertama pada papan sesuai prioritas
2. Cek untuk setiap kemungkinan gerakan (atas, bawah, kiri, kanan) apakah gerakan yang dilakukan memenuhi fungsi pembatas atau tidak
3. Jika memenuhi, lakukan gerakan pada permainan, ekspansi simpul sekarang menjadi simpul yang menandakan langkah selanjutnya sebagai status permainan selanjutnya secara DFS. Jika tidak memenuhi, pangkas simpul tersebut, kembali ke langkah 2

4. Untuk simpul yang diekspansi, ulangi mulai dari langkah 2 hingga terdapat solusi yang memenuhi persoalan (mendapat kotak dengan label angka tertentu). Jika tidak terdapat solusi dari simpul tersebut, *backtrack* dan ulangi ekspansi dari simpul yang masih memiliki anak yang belum diekspansi.
5. Untuk simpul yang memenuhi persoalan, catat status akhir dan urutan langkah untuk mencapai solusi tersebut.

IV. PENGUJIAN

A. *Pengujian Algoritme*

Untuk mempermudah penggambaran pohon ruang status, permainan pada pengujian ini akan dibatasi lebih lanjut dengan grid permainan berukuran 2×2 , dan kotak tujuan bernilai 4, yang secara teori dapat diselesaikan hanya dengan satu langkah.

Mula-mula, inisialisasi permainan dengan status awal yang digambarkan pada gambar berikut.

Gambar 10. Status awal pengujian.
Sumber : dokumen penulis

Langkah-langkah penyelesaian persoalan di atas menggunakan algoritme *backtracking* adalah sebagai berikut.

1. Ekspansi status awal dengan mencoba menggerakkan permainan ke atas, kiri, kanan, dan bawah.

   Uji atas : ketika diuji pada fungsi pembatas, pergerakan ke atas tidak menghasilkan perubahan status permainan, sehingga simpul anak ini dimatikan.

   Uji kiri : ketika diuji, ternyata pergerakan ke kiri menghasilkan perubahan status permainan, juga menghasilkan konfigurasi yang mengandung kotak tujuan, yaitu kotak berlabel 4. Simpul dimatikan dan dimasukkan ke dalam himpunan solusi.

   Uji bawah : ketika diuji, ternyata pergerakan ke kiri menghasilkan perubahan status permainan, namun belum mengandung kotak berlabel 4. Simpul diekspansi untuk dieksplor lebih lanjut.

2. Ekspansi simpul hasil pergerakan ke bawah dari simpul status awal, dengan mencoba menggerakkan permainan ke atas, kiri, kanan, dan bawah.

   Uji atas : ketika diuji, ternyata pergerakan ke atas menghasilkan perubahan status permainan, juga menghasilkan konfigurasi yang mengandung kotak tujuan, yaitu kotak berlabel 4. Simpul dimatikan dan dimasukkan ke dalam himpunan solusi.

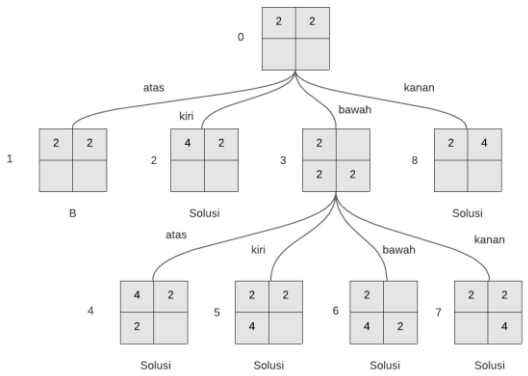
   Uji kiri : ketika diuji, ternyata pergerakan ke kiri menghasilkan perubahan status permainan, juga menghasilkan konfigurasi yang mengandung kotak tujuan, yaitu kotak berlabel 4. Simpul dimatikan dan dimasukkan ke dalam himpunan solusi.

   Uji bawah : ketika diuji, ternyata pergerakan ke bawah menghasilkan perubahan status permainan, juga menghasilkan konfigurasi yang mengandung kotak tujuan, yaitu kotak berlabel 4. Simpul dimatikan dan dimasukkan ke dalam himpunan solusi.

   Uji kanan : ketika diuji, ternyata pergerakan ke kanan menghasilkan perubahan status permainan, juga menghasilkan konfigurasi yang mengandung kotak tujuan, yaitu kotak berlabel 4. Simpul dimatikan dan dimasukkan ke dalam himpunan solusi.

3. *Backtrack* ke simpul yang masih memiliki kandidat simpul anak dan masih bisa diekspansi (belum mati), yaitu simpul status awal (akar). Karena uji atas, kiri, dan bawah sudah dilakukan, maka akan dilakukan uji kanan yang menghasilkan perubahan status permainan, juga menghasilkan konfigurasi yang mengandung kotak tujuan, yaitu kotak berlabel 4. Simpul dimatikan dan dimasukkan ke dalam himpunan solusi.
4. Semua simpul hidup sudah tidak memiliki kandidat simpul anak lagi, sehingga proses pencarian solusi dihentikan.

Pohon ruang status yang terbentuk dari langkah-langkah yang telah dituliskan dapat diilustrasikan lewat gambar berikut.

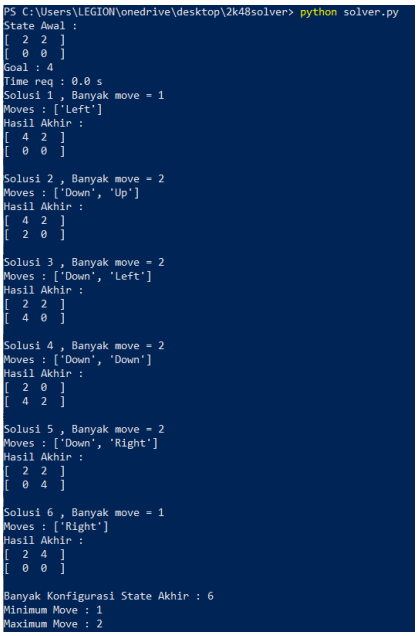


Gambar 11. Pembangkitan Pohon Ruang Status.
Sumber : dokumen penulis

B. *Pengujian Implementasi Program*

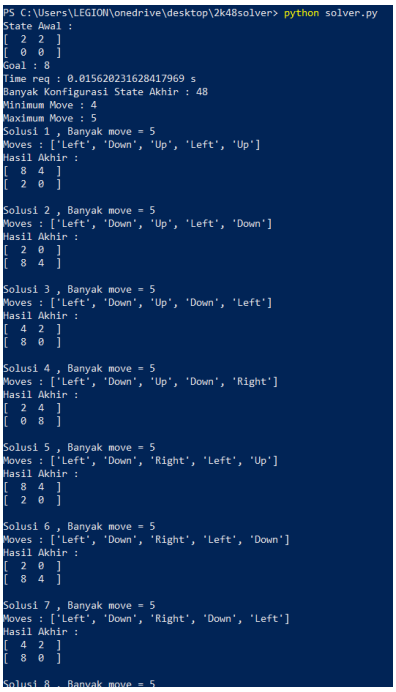
Sebagai referensi, kode program dapat diakses melalui tautan https://github.com/moshval/2k48solver . Pada sub-bab ini akan dilakukan pengujian terhadap program yang sudah dibuat untuk menyelesaikan permainan 2048 deterministik menggunakan algoritme *backtracking*.

1. Pengujian Program untuk Kasus grid berukuran 2×2 , dan kotak tujuan bernilai 4 (sama seperti pada bagian A)



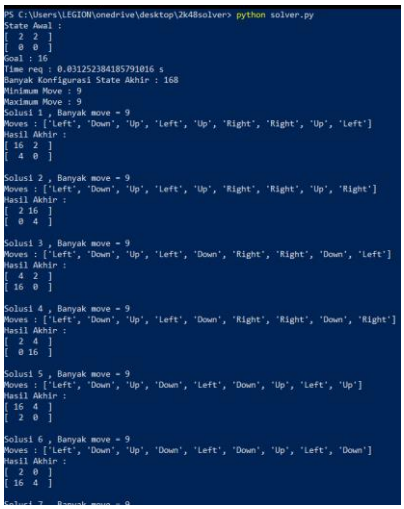
Gambar 12. Pengujian 1
Sumber : dokumen penulis

2. Pengujian Program untuk Kasus grid berukuran 2×2 , dan kotak tujuan bernilai 8



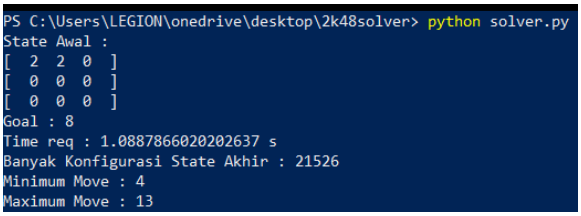
Gambar 13. Pengujian 2. Tidak semua solusi dicetak (keterbatasan ruang)
Sumber : dokumen penulis

3. Pengujian Program untuk Kasus grid berukuran 2×2 , dan kotak tujuan bernilai 16



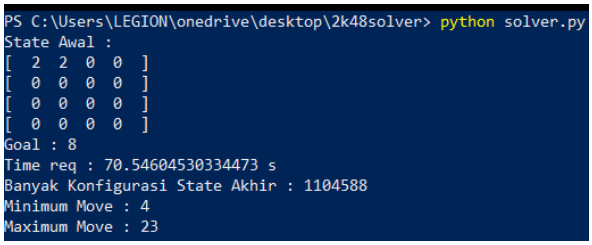
Gambar 14. Pengujian 3
Sumber : dokumen penulis

4. Pengujian Program untuk Kasus grid berukuran 3×3 , dan kotak tujuan bernilai 8 (solusi per daun tidak ditulis)



Gambar 15. Pengujian 4
Sumber : dokumen penulis

5. Pengujian Program untuk Kasus grid berukuran 4×4 , dan kotak tujuan bernilai 8 (solusi per daun tidak ditulis)



Gambar 16. Pengujian 5
Sumber : dokumen penulis

Untuk pengujian-pengujian dengan ukuran grid yang lebih besar dan kotak tujuan yang bernilai lebih besar akan dibutuhkan lebih banyak langkah, juga membuat lebih banyak kombinasi konfigurasi yang terbuat dari langkah-langkah tersebut, menyebabkan waktu yang dibutuhkan untuk mendapatkan *state* akhir lebih lama, pada pengujian 5 dibutuhkan 70.5 detik untuk mendapatkan semua solusi pada kasus grid berukuran 4×4 dan kotak tujuan bernilai 8. Penggunaan algoritme *backtracking* mampu untuk mengurangi jumlah waktu yang dibutuhkan namun pada kasus ini terlalu banyak komputasi dan rekursi yang dibutuhkan sehingga tidak dapat mengakomodasi persoalan yang lebih besar.

V. Simpulan

Permainan 2048 merupakan puzzle atau permainan teka-teki berbasis kotak geser (*sliding tile puzzle*) dimainkan dalam mode *single-player* (memiliki pemain tunggal), di mana pemain bermain dengan menggeser-geser kotak pada grid berukuran 4×4 untuk mendapat angka 2048. Solusi untuk permainan ini dapat dimodelkan menggunakan algoritme *backtracking* yang dapat memangkas simpul status yang sekiranya tidak mengarah ke tujuan.

Penggunaan algoritme *backtracking* pada persoalan ini mampu menyelesaikan persoalan permainan 2048 dengan ukuran yang relatif kecil, karena untuk ukuran permainan yang cukup besar maka akan dibutuhkan waktu yang cukup lama untuk mencapai solusi, atau bahkan menimbulkan status run berupa *time limit exceeded* (TLE). Hal ini mungkin disebabkan karena banyaknya *state* yang harus diekspansi dan diproses, menimbulkan rekursi yang banyak, atau karena buruknya desain kode permainan yang penulis buat.

Tautan Video dan Kode Program

Video : https://youtu.be/0ZhhBnPepjU

Kode Program : https://github.com/moshval/2k48solver

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Tuhan YME karena atas karunia-Nya sehingga penulis dapat menyelesaikan makalah ini tepat pada waktu yang ditetapkan. Juga terima kasih kepada seluruh pihak yang telah membantu pembuatan makalah ini, baik secara langsung maupun tidak langsung. Secara khusus, penulis mengucapkan terima kasih sebesar-besarnya atas bimbingan Bapak dan Ibu dosen mata kuliah IF2211 Strategi Algoritma selama semester ini..

Referensi

[1] https://kbbi.web.id/algoritme. Diakses pada 9 Mei 2021.

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf. Diakses pada 10 Mei 2021

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf. Diakses pada 11 Mei 2021

[4] https://www.businessinsider.com/2048-puzzle-game-2014-3?r=US&IR=T . Diakses pada 11 Mei 2021

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Tangerang, 8 Mei 2021

Mohammad Sheva Almeyda Sofjan
13519018